Sci

MDPI

Article

QuantumGS-Box—A Key-Dependent GA and QRNG-Based S-Box for High-Speed Cloud-Based Storage Encryption

Anish Saini *, Athanasios Tsokanos and Raimund Kirner

School of Physics, Engineering and Computer Science (SPECS), University of Hertfordshire, Hertfordshire AL10 9AB, UK; a.tsokanos@herts.ac.uk (A.T.); r.kirner@herts.ac.uk (R.K.)
* Correspondence: a.saini@herts.ac.uk

Abstract: Cloud computing has revolutionized the digital era by providing a more efficient, scalable, and cost-effective infrastructure. Secure systems that encrypt and protect data before it is transmitted over a network and stored in the cloud benefit the entire transmission process. Transmission data can be encrypted and protected with a secure dynamic substitution box (S-box). In this paper, we propose the QuantumGS-box, which is a dynamic S-box for high-speed cloud-based storage encryption generated by bit shuffling with a genetic algorithm and a quantum random number generator (QRNG). The proposed work generates the S-box optimized values in a dynamic way, and an experimental evaluation of the proposed S-box method has been conducted using several cryptographic criteria, including bit independence criteria, speed, non-linearity, differential and linear approximation probabilities, strict avalanche criteria and balanced output. The results demonstrate that the QuantumGS-box can enhance robustness, is resilient to differential and provide improved linear cryptoanalysis compared to other research works while assuring non-linearity. The characteristics of the proposed S-box are compared with other state of the art S-boxes to validate its performance. These characteristics indicate that the QuantumGS-box is a promising candidate for cloud-based storage encryption applications.

Keywords: dynamic S-box; QRNG; cloud-based storage; encryption

Citation: Saini, A.; Tsokanos, A.; Kirner, R. QuantumGS-Box—A Key-Dependent GA and QRNG-Based S-Box for High-Speed Cloud-Based Storage Encryption. *Sci* **2024**, *6*, 86. https://doi.org/10.3390/sci6040086

Academic Editor: Luis Javier Garcia Villalba

Received: 13 October 2024
Revised: 8 December 2024
Accepted: 12 December 2024
Published: 23 December 2024

1. Introduction

Cloud computing has become a significant part of the IT landscape in the last few years due to the migration from local drive storage to cloud computing storage. This leads to data being stored remotely rather than locally and with interconnected networks, being accessed over the Internet [1]. Transmission of data over a network requires the consideration of a number of factors, including the processing speed and security. Data security is a significant concern as information flows through the internet. Incorporating cryptography [2] that meets secure transmission requirements ensures that the data is transmitted securely. Secure data transmission over a network transmission can be achieved by cryptography that uses *Key Schedule Algorithms* (KSA) [3] and *encryption* [4].

KSA includes methods of generating a random key typically based on substitution and permutation. The substitution replaces the specific bits with other defined bits, and a *substitution-box* (S-box) is used to implement substitution. Permutation includes the shuffling of bits with specific operations. The S-box performs substitution in both KSA and encryption.

Our approach is shown in Figure 1, where Alice is both the sender and receiver. Alice encrypts the data before sending it to the network receiving the same encrypted data. Encrypting the data before sending it to the network is the most effective way of protecting cloud-based (remote) storage from unauthorised access or attacks.

Figure 2 illustrates a cloud-based storage network that uses a static S-box to implement KSA and encryption.

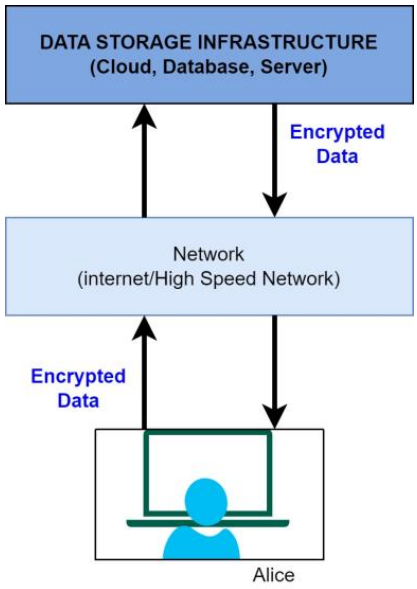
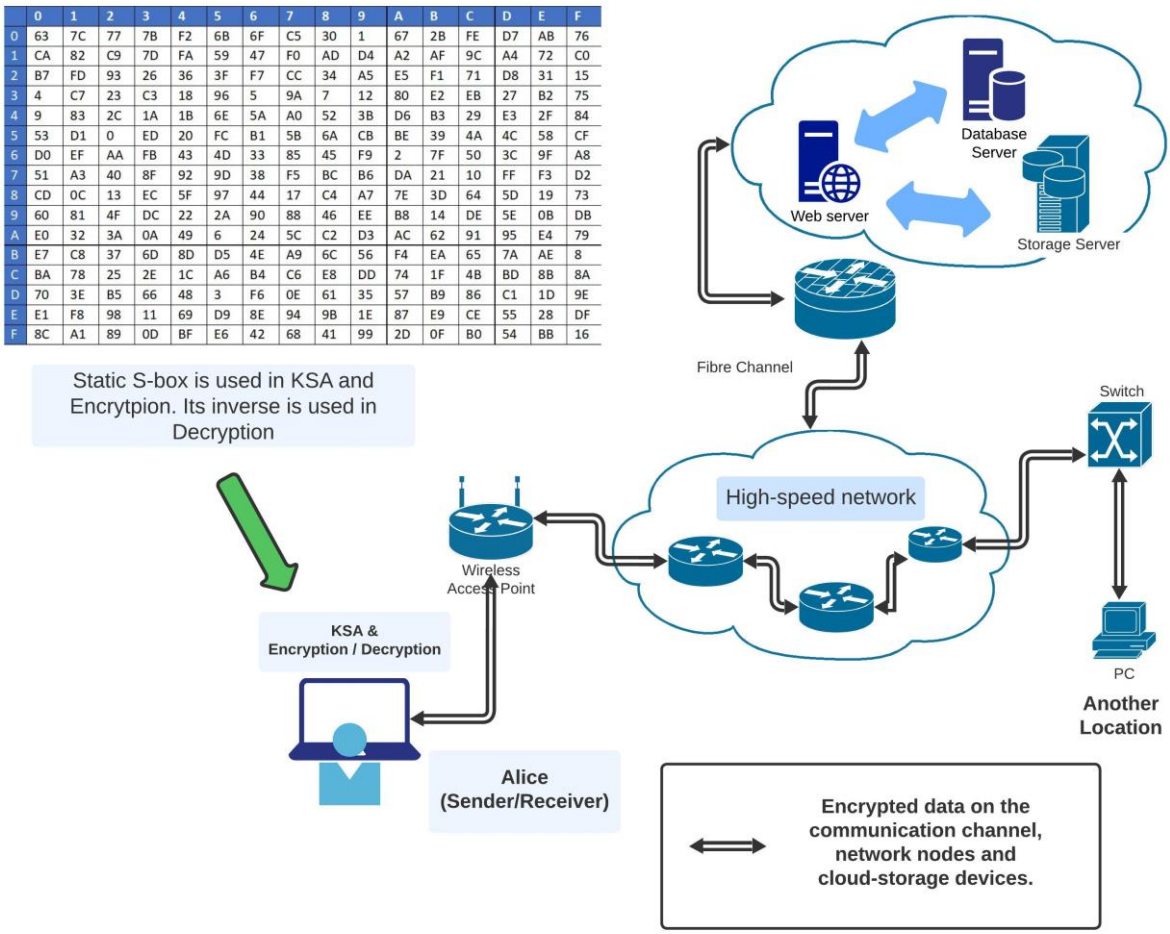


Figure 1. Alice is sender and receiver.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	1	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	4	C7	23	C3	18	96	5	9A	7	12	80	E2	EB	27	B2	75
4	9	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	0	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	2	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	6	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	8
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	3	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Figure 2. Usage Motivation: AES static S-box in cloud-based storage encryption through high-speed network.

The AES S-box shown is a 16 × 16 matrix of static bits. Using the KSA and S-box, Alice generates the key and encrypts the data. Alice encrypts the data before sending it

to cloud-based storage (remote) through HSN, and then decrypts it after receiving it. The inverse of an S-box is used to decrypt data.

The static S-box is used to provide a substitution of values. The security strength of the system depends on how vulnerable is the key generated by the KSA and its components [5] for the cryptanalysis attack, which is a key point of our research. In the last two decades the concept of a dynamic S-box has become an important research area for enhancing security. The static bits of an S-box make it vulnerable to attack and many researchers have focused on making a dynamic S-box with different scientific approaches.

There are various approaches of creating dynamic S-box with

- *Algebraic theory-based*: A mixture of classical and modern mathematics, Galois Theory processed the solutions of polynomial equations (mathematical equations made of numbers and variables) to form different values of dynamic S-box [6–9].
- *Chaos theory-based:* The theory uses different patterns and mathematical formulations to generate randomness in the system and corresponds to different dynamic values of the S-box [10–12].
- *Random number generator-based*—The random number generated by sources like electrical and quantum noise leads to pseudo, true, and quantum generators, processing the values of dynamic S-box [13,14].
- *Advanced data analysis techniques-based:* Advanced data analysis techniques like machine learning, particle swarm optimization, genetic algorithms, and heuristic techniques analyze the different sources of data and generate the values of the S-box dynamically [15–17].

Conventional AES keys are based on Pseudo-Random Number Generators [18,19] to generate random data to enhance AES's security. RK-AES [20] proposed a Symmetric Random Function Generator to achieve randomness in AES's key. However, the most unpredictable [21] and truly random data [22] can be achieved using a *quantum random number generator* (QRNG) [23].

Miguel Herrero-Collantes et al. explained different types of Quatum Number Generator based on different sources like optical and non-optical QRNG [24]. Xiongfeng Ma et al. categorized the QRNGs into three groups: the practical QRNG (generate randomness at a high speed and built on fully trusted and calibrated devices), self-testing QRNG (randomness without the trusting the actual implementation) and semi-self-testing QRNG (a tradeoff between the trustworthiness on the device and the random number generation speed) [25]. Jinlu Liu et al. proposed a 117 Gbits/s random bit generation rate QRNG using min-entropy estimation and Toeplitz-hashing randomness extraction, based on the quantum phase fluctuation of a distributed feedback (DFB) laser [26]. Meilana Siswanto, et al. designed a photonic-based RNG to enhance the security of an Internet of Things (IoT) system comprised of optical components, analog-digital electronic systems, and an asynchronous transmitter [27]. It also utilizes Verilog firmware to integrate the IoT system. Zhu Cao et al. proposed a source-independent QRNG in which output randomness can be certified even when the source is uncharacterized and untrusted. Figure 3 shows the random number generation process using optical device and generating quantum random numbers.

IDQ created a hardware QRNG for generating true (non-pseudo) randomness. Quantis [22] is a state-of-the-art QRNG, exploiting an optical quantum process as the source of randomness. A QRNG is superior to traditional random number generators as their source of randomness is associated with environmental perturbations [28]. The device has also a monitoring function. If a failure is detected, the random bit stream created is immediately disabled. The device provides true randomness from the first bit, separates Quantum noise from classical noise, auto calibrated and certified [29].

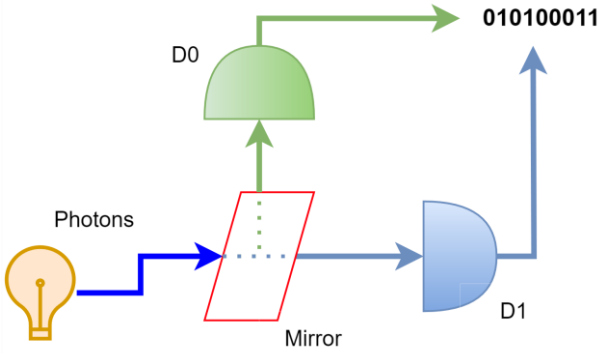


Figure 3. Random number generation using optical process as QRNG.

On the other hand, a genetic algorithm [30] is a search heuristic inspired by the natural evolution of Charles Darwin's theory. The algorithm corresponds to the natural selection of selecting the fittest individuals based on a fitness function in order for reproduction to produce offsprings of the next generation. A genetic algorithm requires the following two factors of a domain solution: a genetic representation and a fitness function to evaluate suitability. Initial population, fitness function, selection, crossover, mutation, and termination are phases of a genetic algorithm aiming to an optimal solution.

Sourabh Katoch et al. reviewed various advantages, disadvantages of different GAs like Classical GA, Binary coded GAs, Real-coded GAs, Multiobjective GAs, Parallel GAs, Chaotic GAs and Hybrid GAs [31]. They also discussed various operators used for different GAs. Yong Wang et al. proposed a novel method for constructing S-box by transforming it to a Traveling Salesman Problem and designed S-box based on chaos and genetic algorithm [32]. Another research by Yong Wang et al. proposed a novel genetic algorithm to construct bijective S-boxes with high nonlinearity, taking the nonlinearity of the S-box as the optimization objective [33].

Our research focuses on improving security by creating an S-box that is dynamic using a quantum random number generator and evolutionary computation. The KSA depends on a proposed dynamic S-box based on quantum random bits and a genetic algorithm in order to generate the key.

Our proposed work uses a 128-bits secret key and 128-bits from the QRNG as an initial population and performs crossover and mutation functions to generate different values in the S-box.

The focus of this article is to create a key-dependent dynamic S-box based on quantum randomness and a bit shuffling method. The contributions of this work are:

- Designing an algorithm that takes a user-defined key and quantum random bits from a QRNG and generates a dynamic S-box using bit shuffling with genetic evolution.
- Analysing the proposed algorithm corresponding to a range of cryptographic properties, including nonlinearity, randomness, linear and differential probabilities, bit independence criteria, and balance.

The organisation of this paper is as follows: Section 2 covers the background of AES static S-boxes; Section 3 provides information about related work; Section 4 introduces the design principles and proposed methodology of the proposed dynamic S-box (QuantumGS-box); The test results and analysis are presented in Section 5, while Section 6 concludes the paper.

2. Background

An AES static S-box [34,35] having the order $(p \times q)$ is a mapping function $L = S(c)$, where $S = \{0,1\}^p \rightarrow \{0,1\}^q$ which is used to map p-bits input string c to q-bits output string L. The input is mapped with its multiplicative inverse in $GF(2)^8$.

I. The first step is a nonlinear function $f(c)$ defined as:

$$f(c) = \begin{cases} 0, if\ c = 0 \\ c^{-1}, if\ c \neq 0 \end{cases} \quad (1)$$

The function $f(c)$ maps zero to zero, and for a non-zero field element c, it maps the element to its multiplicative inverse c^{-1} in $GF(2)^8$

II. The multiplicative inverse is then transformed using the following affine transformation

$$\mathrm{L} = \mathrm{Affine}_8(\mathrm{C}) + \mathrm{b} \quad (2)$$

where Affine_8 is 8×8 matrix and b is a constant. Namely, for a field element C = (C7, C6, C5, C4, C3, C2, C1, C0), L = Affine_8(C) + b with

$$\begin{bmatrix} L_7 \\ L_6 \\ L_5 \\ L_4 \\ L_3 \\ L_2 \\ L_1 \\ L_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} c_7 \\ c_6 \\ c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Table 1 shows an AES static S-box. It is a 16×16 matrix of hexadecimal values that are used in KSA and encryption. There are several cryptographic properties that make an AES S-box unique, such as high nonlinearity, correlation immunity, algebraic immunity, and no fixed points or opposite fixed points. However, because of its static nature, it is susceptible to enhanced cryptoanalysis techniques [36,37].

Table 1. Value matrix of an AES Static S-box.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	1	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	4	C7	23	C3	18	96	5	9A	7	12	80	E2	EB	27	B2	75
4	9	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	0	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	2	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	B	DB
A	E0	32	3A	A	49	6	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	8
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	3	F6	E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	D	BF	E6	42	68	41	99	2D	F	B0	54	BB	16

The other part of this research study is the evolution computation with a *genetic algorithm*. A *genetic algorithm* (GA) is a well-known algorithm for search-oriented optimization inspired by biological evolution. A GA derives from the Darwinian theory of survival of the fittest. A GA dynamically changes the search process to reach an optimal solution based on crossover and mutation probabilities as a part of different phases of its cycle. Binary GA, as a search-oriented optimization, dynamically searches the solution within the generations

of the population encoded in the binary format. The fitness function is used to evaluate the population that undergoes a crossover or mutation to find the best value.

Our proposed work uses the transformation operations of a binary GA to provide efficient shuffling for the determination of the values of a dynamic S-box.

The initial population phase sets the foundation for the genetic algorithm by creating a diverse set of candidate solutions. The fitness function phase evaluates the quality of each solution, allowing the algorithm to prioritize the fittest individuals. The selection phase determines which individuals will be chosen as parents for reproduction, promoting the propagation of desirable traits. The crossover phase combines the genetic information of selected parents to create offspring with potentially improved characteristics. The mutation phase introduces small random changes to the offspring, ensuring the exploration of new solutions. Finally, the termination phase stops the algorithm once a satisfactory solution is found, or a predefined number of generations is reached.

3. Related Work

During the past two decades, cryptosystem designers have become increasingly concerned with designing key-dependent S-box methods with random properties. Various cryptoanalysis techniques, such as differential, linear, meet-in-the-middle, key recovery, and shortcut attacks, require dynamic methods and randomly generated substitution cryptosystems. As such, a strong emphasis has been placed on developing and implementing key-dependent S-box methods with random properties resistant to various cryptoanalysis techniques. Julia Juremi et al. surveyed many dynamic S-boxes focusing on framing the technology for generating them and creating them dynamically to increase cryptographic strengths and resist different attacks significantly [38].

Various methodologies for constructing dynamic S-boxes have been proposed, including pseudo random number generators (PRNGs) and true random number generators (TRNGs) with specific considerations, Chaos-based, algebraic, and optimization techniques.

Mangal Deep Gupta et al. proposed an architecture for reconfigurable PRNG designed using Verilog HDL, synthesized on the Xilinx tool using the Virtex-5 (XC5VLX50T) and Zynq (XC7Z045) FPGA [39]. Utilizing these PRNGs, they design two 16×16 substitution boxes (S boxes). The proposed S-boxes fulfil the cryptographic criteria such as Bijective, Balanced, Nonlinearity, Dynamic Distance, Strict Avalanche Criterion (SAC), and BIC nonlinearity criterion. They enhanced the security for image encryption by using these two S-boxes in terms of various parameters of image encryption like key space, information entropy, number of pixels change rate (NPCR), unified average changing intensity (UACI), and image encryption quantitatively in terms of (mean-squared error (MSE), structural similarity index (SSIM) and peak signal-to-noise ratio (PSNR)). Mengdi Zhao et al. a non-degenerate 2D enhanced quadratic map (2D-EQM) that exhibits ergodicity and randomness. In addition, they used it to generate affine transformation matrices and constants for seeding S-boxes based on affine transformation matrices [40]. They generated a number of keyed strong S-boxes with high nonlinearity.

Researchers have also used irreducible polynomials, additive constants, and static lookup tables [41] to make a dynamic S-box. Alamsyah et al. proposed a combination of an irreducible polynomial (irreducible polynomials) and an affine matrix (a Boolean operation XOR for each affine matrix element of the AES S-box) [42]. Several high-quality S-boxes with corresponding cryptographic properties are built from the combination of the results of these modifications. Using dynamic S-boxes and shift rows to simplify the encryption process also increased the encryption strength of the AES cipher [43]. Manjula G. et al. proposed that the modified S-boxes are dynamic, random, and key-dependent, adding to the algorithm's complexity and making cracking them more challenging [44]. Praveen et al. proposed a dynamic key-dependent S-box that relies on affine transformations with irreducible polynomials and affine constants [45]. Grasha Jacob et al. developed an S-box that can be generated dynamically based on the sub-byte transformation used in cryptography [46].

Chaos-based dynamic S-boxes are also of interest to many researchers. Alaa F. Kadhim et al. used shifting, chaotic theory (1D, 2D logistic maps), and particle swarm algorithms to generate a dynamic S-box based on the input key [17]. The system provides enhanced cryptographic properties. The authors of another study proposed an uncorrelated S-box element for generating an S-box that meets cryptographic criteria like bijection, nonlinearity, strict avalanche, output bit independence, equiprobable input/output XOR distribution, and maximum expected linear probability computed on typical chaos-based schemes without taking into account the lag time of a chaotic series [47]. Muhammad et al. report a new chaos-based affine transformation generation method that uses rotational matrices to generate key-based S-boxes [48].

Researchers also focus on genetic algorithms for creating dynamic S-boxes that lead to secure symmetric cryptosystems. Aguirre et al. proposed high nonlinearity by using the multi-objective evolutionary approach to evolve Boolean functions [49]. Stjepan Picek et al. experimented with a genetic algorithm and programming by modifying the mutation operator and initialization process to search Boolean functions with cryptographic properties [50]. Anand Kumar et al. proposed a robust S-box design with a hybrid approach of GA and Particle Swarm Optimization (PSO) [51]. Their performance evaluation is better regarding nonlinearity, strict avalanche effects, bits distribution, and bijectivity. By employing a GA in SP boxes and implementing a nonlinear neural network in the SP network, Kalaisel et al. proposed a redesigned, enhanced AES cryptosystem that increases security against timing attacks and reduces computation time. In order to reduce the probability of the algorithm being impervious to future dialects, Alaa F. Haitham et al. used the chaotic function to generate an initial random sequence of bits and the quantum crossover to provide a new and improved substitution box with increased nonlinearity [52].

Researchers focused on creating dynamic S-boxes; however, generating dynamicity using quantum randomness and a user key and creating a dynamic S-box while retaining the box's cryptographic properties is still an unexplored of research.

Our proposed substitution method contributes to existing knowledge by enhancing security of symmetric cryptosystems in HSN by generating the values of S-box at the time of execution from the secret key and (QRNG).

4. QuantumGS-Box—High-Speed Cloud-Based Storage Encryption with GA-Based Key Optimisation

The proposed method of generating a dynamic S-box includes a QRNG and bit shuffling based on the operations of a genetic algorithm to make the mathematical calculation simple while at the same time ensuring that the values generated by this method are highly secure. The proposed QuantumGS-box values are generated dynamically with the user defined key and quantum random number generated bits. The design principle is discussed first, followed by the proposed methodology.

4.1. The Design Principle

The design principle of the proposed work uses algebraic techniques, quantum randomness, and optimization techniques to generate a dynamic, highly secure S-box. There are a variety of cryptographic properties associated with a highly secure S-box. These properties include nonlinearity, balance, correlation immunity, algebraic degree, and differential and linear approximation probabilities. A diagram illustrating the design principles is shown in Figure 4.

The *random number generators* (RNGs)-based S-box effectively affects the randomness of bits in the S-box. Pseudo-random number generator (PRNG) [39] and true random number generator (TRNG) [13] based S-boxes are proposed by some researchers in order to enhance the cryptographic properties. However, QRNG [53] ensures high entropy using quantum states of light. Quantis [22], a QRNG developed by IDQ, generates random numbers using a quantum process [54,55]. The device allows live verification of its operation and provides the highest level of entropy without requiring a post-processing function

to increase its entropy rate. QRNGs [23] are considered superior to traditional random number generators, as their source of randomness is invulnerable to environmental perturbations such as temperature, voltage, or current and are provably secure random number generators [28,56].

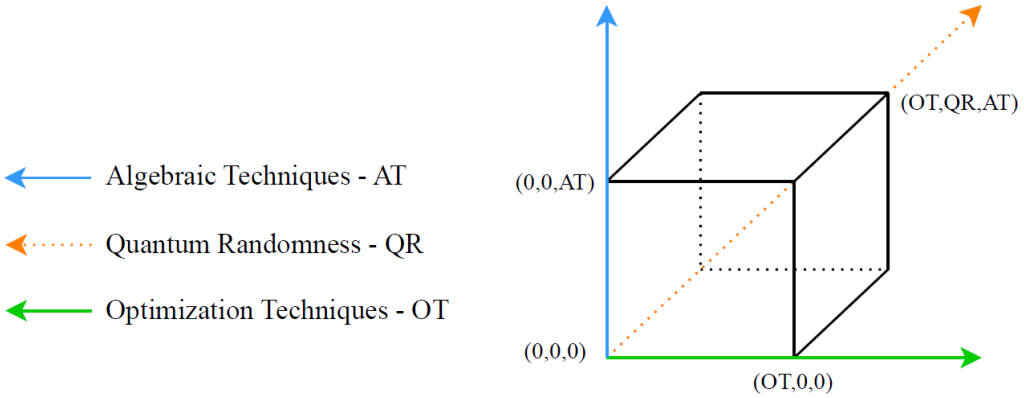


Figure 4. Design principle of the proposed work.

Algebraic methods are used to achieve various cryptographic properties in generating the S-box. However, with the advancement of cryptanalysis [57,58] over the last few years, these methods are becoming more vulnerable to exploitation.

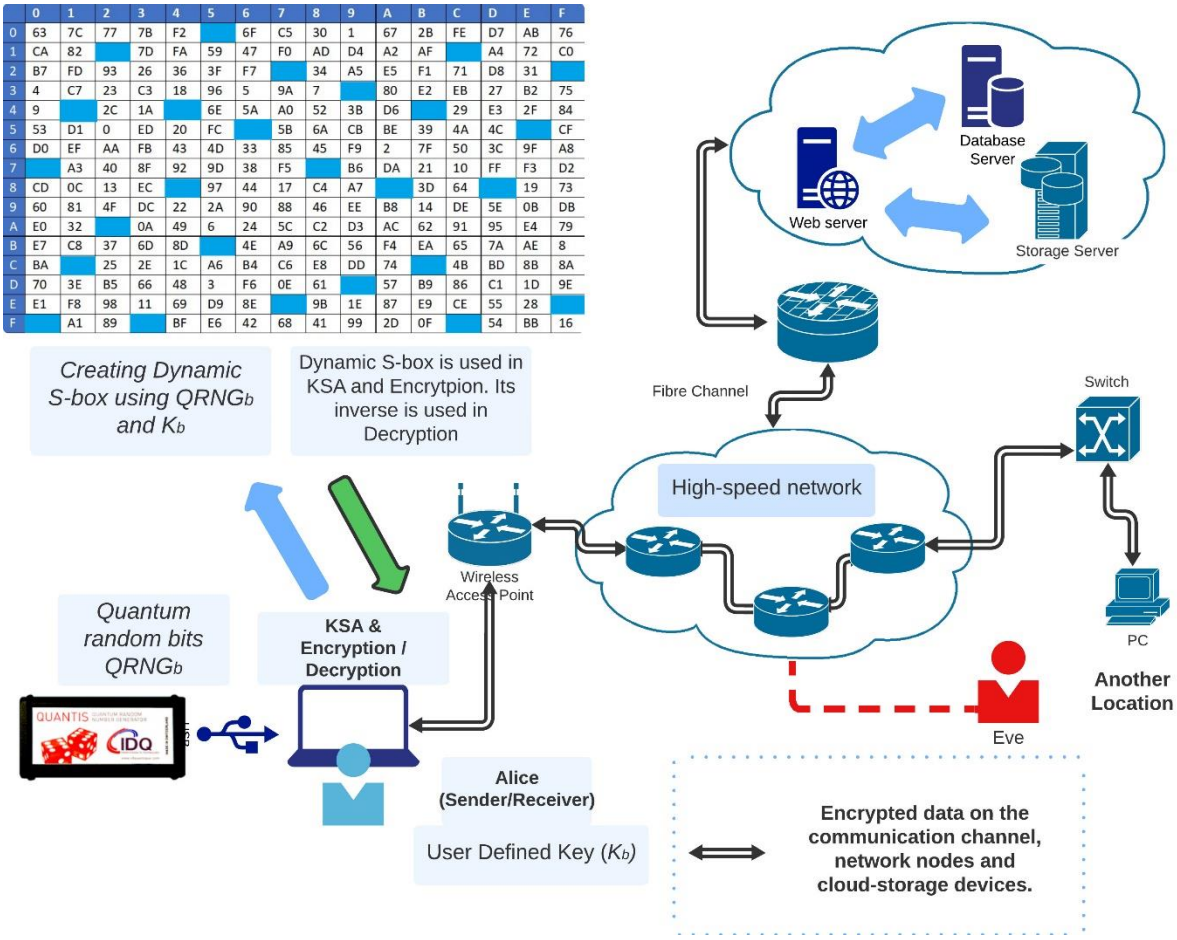
Another important criterion in the design of S-box dynamic bits is the speed at which the bits are generated. There needs to be an increase in the speed at which the S-box is constructed to make it more difficult to access the bits in the S-box. Dynamic S-boxes comprise the basis for generating the key and encrypting the data using that key. The higher speed of generating dynamic S-box values also contributes to the high-speed encryption of the data. Increasing the generation speed of dynamic S-box values reduces encryption time and data travel time on communication channels and data storage facilities.

The last criterion is the optimal values based on all three above-mentioned criteria, which can be achieved through evolutionary computation (EC). There are various studies [59–61] by researchers who created dynamic S-boxes based on EC and optimized the most effective solution for creating S-box values.

4.2. Proposed Methodology

Figure 5 shows the dynamic proposed S-box, called QuantumGS-box, for KSA and for a cloud-based storage encryption. The figure shows the encryption with the new QuantumGS-box. The QuantumGS-box is a 16×16 matrix consisting of dynamic bits. The user will generate the key with KSA along with S-box and encrypt the information. Alice is both sender and receiver in this scenario, who encrypts the data before sending it to the high-speed network and decrypt it once it receives back.

The QuantumGS-box provides a matrix of substitution values based on a QRNG and a user defined key for KSA and encrypting the data. Alice will send the encrypted information over a communication channel to the cloud. The encrypted format makes the data secure before traveling through various HSN nodes. This ensures that the data is protected from Eve, a potential malicious attacker, who could attempt to intercept the data while in transit. The encrypted data is stored on multiple servers depending on their cloud application. The same encrypted data travels back over the network when Alice wants to access the data. Overall, this process ensures that Alice's data remains secure and confidential during data transmission to the cloud.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2		6F	C5	30	1	67	2B	FE	D7	AB	76
1	CA	82		7D	FA	59	47	F0	AD	D4	A2	AF		A4	72	C0
2	B7	FD	93	26	36	3F	F7		34	A5	E5	F1	71	D8	31	
3	4	C7	23	C3	18	96	5	9A	7		80	E2	EB	27	B2	75
4	9		2C	1A		6E	5A	A0	52	3B	D6		29	E3	2F	84
5	53	D1	0	ED	20	FC		5B	6A	CB	BE	39	4A	4C		CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	2	7F	50	3C	9F	A8
7		A3	40	8F	92	9D	38	F5		B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC		97	44	17	C4	A7		3D	64		19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32		0A	49	6	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D		4E	A9	6C	56	F4	EA	65	7A	AE	8
C	BA		25	2E	1C	A6	B4	C6	E8	DD	74		4B	BD	8B	8A
D	70	3E	B5	66	48	3	F6	0E	61		57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E		9B	1E	87	E9	CE	55	28	
F		A1	89		BF	E6	42	68	41	99	2D	0F		54	BB	16

Figure 5. Proposed QuantumGS-box in cloud-based storage encryption through high-speed network.

4.2.1. Step by Step Procedure and Algorithm to Generate QuantumGS-Box

This section illustrates the mathematical calculation and the bit shuffling based on the operations of a *genetic algorithm* (GA) of the proposed method to generate dynamic S-box values. The step-by-step operation of the proposed method is shown with a control-flow diagram in Figure 6.

The flowchart starts with taking a user key from the user and then random bits from QRNG. The genetic algorithm will be processed with initial population, the generation and generating the chromosomes. A fitness function is used to select the best chromosomes in order to find the index to generate the values of QuantumGS-box.

Step1 (S1): Initially, the user inputs a 16-character key, denoted as $UKey$
Step2 (S2): The QRNG generates the 128 quantum random bits, denoted as $QRNG(128)$
Step3 (S3): The user key UKey is converted into bits and denoted as Key_b
Step4 (S4): Initial Population generation (P_{init})

- $QRNG(128)$—Quantum random bits from QRNG (128 bits)
- Key_b —User defined key (128 bits)

$$P = P \cup (QRNG(128) \oplus Key_b) \tag{3}$$

Step5 (S5): This step calls the genetic algorithm with the function name optimize_GA and a parameter initial population (P_{init}), calculated at S4. Algorithm 1 shows the pseudo-code of the proposed algorithm for optimize_GA. Section 4.2.2 focuses on the optimize_GA.

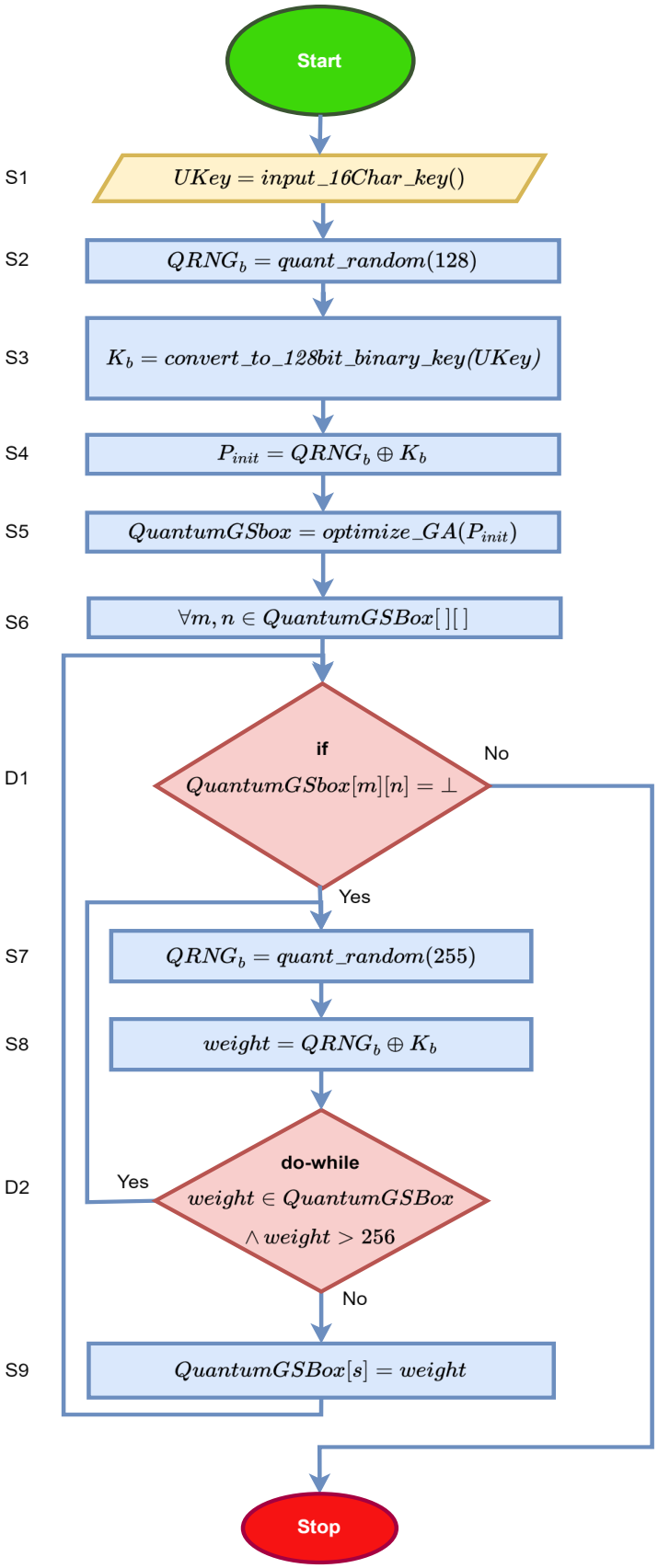


Figure 6. Flowchart of the proposed QuantumGS-box.

Step 6 (S6): This step checks each element in the $QGSbox$ with row m and column n. The decision depends on the positions that have NaN ($\perp$) as their value.

The following steps will calculate the weight of that positions, where there is a NaN (Not a Number) in the $QGSbox$. If there is no NaN, then this will proceed to stop.

Step 7 (S7) to Step 9 (S9): These steps will continue till $QGSbox$ has NaN ($\perp$).

The quantum random number bits will be generated from QRNG and as $QRNG(255)$

The value of temporary weight denoted as *weight* is calculated by using a XOR function on the quantum random bits and user-defined key bits.

$$weight = QRNG(255) \oplus Key_b \tag{4}$$

The next block will check if the value for *weight* has already been stored in $QGSbox$, and greater than 256. If it is, the value will be regenerated; if it does not, the value will be stored in the $QGSbox$.

When the algorithm reaches the stop, then all $QGSbox$ values have been calculated and the algorithm terminates. Algorithm 2 shows the pseudo-code of the proposed algorithm.

Algorithm 1: Algorithm $optimize\_GA(P_{init})$: Calculate positions for weights of QuantumGS-box

INPUT: P_{init} *//initial population (with PSIZEchromosomes)*

$SSbox = \begin{pmatrix} a_{00} & a_{01} & \cdots & a_{0F} \\ a_{01} & a_{11} & \cdots & a_{1F} \\ \vdots & \vdots & \ddots & \vdots \\ a_{F0} & a_{F1} & \cdots & a_{FF} \end{pmatrix}$ *//static S-Box*

$RMAX$*//max number of S-Box weights to replace:* $10 \leq RMAX \leq 250$

OUTPUT: $QGSbox_{tmp}$*//interim QuantumGS-box*

begin:
  $P_{tmp} = P_{init}$ *//initialize current population with initial population*
  $QGSbox_{tmp} = SSbox$ *//initialize output QuantumGS-box*
  for *gen_cnt* ***is*** 1 ***to*** $RMAX$ *//iterate RMAXtimes*
    $C_{fittest} = \underset{C \in P_{tmp}}{\operatorname{argmax}} fitness\_GA(C)$*//chromosome with max fitness value*
    $i = C_{fittest}[60], \quad j = C_{fittest}[61], \quad k = C_{fittest}[62], \quad l = C_{fittest}[63]$
    $p = C_{fittest}[124], \quad q = C_{fittest}[125], \quad r = C_{fittest}[126], \quad s = C_{fittest}[127]$
    $x = bin\_to\_hex(i, j, k, l)$ // $i \cdot 2^3 + j \cdot 2^2 + k \cdot 2^1 + l \cdot 2^0$
    $y = bin\_to\_hex(p, q, r, s)$ // $p \cdot 2^3 + q \cdot 2^2 + r \cdot 2^1 + s \cdot 2^0$
    $QGSbox_{tmp}[x][y] = \perp$
    $QGSbox_{tmp}[y][x] = \perp$
    $P_{tmp} = mutate(crossover(P_{tmp}))$*//population of next generation*
  return $QuantumGSbox_{tmp}$
end

Algorithm 2: Algorithm to generate QuantumGS-box

INPUT: $UKey$ *//16-character textual key provided by user*

OUTPUT: $QGSbox = \begin{pmatrix} a_{00} & a_{01} & \cdots & a_{0F} \\ a_{01} & a_{11} & \cdots & a_{1F} \\ \vdots & \vdots & \ddots & \vdots \\ a_{F0} & a_{F1} & \cdots & a_{FF} \end{pmatrix}$ *//QuantumGS-box*

begin:
  K_b = *convert_to_128bit_binary_key(UKey)*
  $P_{init} = \varnothing$ *//initial population*
  $\forall i \in \{1 \ldots PSIZE\}.\ P = P \cup (QRNG(128) \oplus Key_b)$ *//fill initial population*

```
5       QGSbox = optimize_GA ( P_init)
6       ∀m, n ∈ QGSBox[ ][ ]
7       If QGSbox[m][n] = ⊥
8           do
9               weight = QRNG(255) ⊕ Key_b
10          while weight ∈ QGSBox ∧ weight > 256
11          QGSbox[s] = weight
12  end
```

Note: ⊥—Not a Number (NaN).

4.2.2. Algorithm $optimize\_GA(P_{init})$

Algorithm 1 shows the steps to calculate the positions for weights of $QGSbox$.

This step will continue to execute from generation 1 to $RMAX$. RMAX is used to define the upper limit of the generation of the genetic algorithm.

In a population of chromosomes, $fitness\_GA(C)$ computes the fitness of each chromosome. The fitness value that is highest across all chromosomes will be considered the most appropriate and its chromosome is denoted as $C_{fittest}$. Algorithms 3 and 4 focuses on $fitness\_GA(C)$ and how to calculate each fitness value.

This highest value chromosome $C_{fittest}$ tells the positions for the weights of $QGSbox$. We have chosen four middle bits and four most significant bits to calculate the row and column index. The index positions a_1 and a_2 represent row and column positions. For example, the 60, 61, 62 and 63 bits of $C_{fittest}$ are 0110 and 124, 125, 127 and 128 bits are 0100, so the $a_1 = 0110$ in binary and $a_1 = 6$ in hexadecimal. Similarly, $a_2 = 0100$ and $a_2 = 4$.

These positions in $QGSbox$ as $QGSbox$ [4][6] and $QGSbox$ [6][4] will now have NaN.

The next step will calculate the next population $\left(P_{tmp}\right)$ using crossover and mutation for the next iteration of the generation.

Crossover: crossover$\left(\mathrm{P_{tmp}}\right)$

We are using the crossover operation of a GA [62]. The chromosomes of each generation are combined with via the crossover function to produce the chromosomes of the next generation. A one-point crossover mechanism has been used. In order to create a new offspring, a crossover point is chosen at random and the tails of $(C_1\ and\ C_2)$ are swapped to create a new offspring. The crossover chromosome is calculated as follows:

$$crossover\left(P_{tmp}\right) = f\left(P_{tmp}\right) \tag{5}$$

where:

$crossover\left(P_{tmp}\right)$—Population chromosome after crossover

n—a random number crossover point derived from the QRNG

Algorithm 3 shows the $crossover\left(P_{tmp}\right)$ function to calculate the crossover for the current population. Algorithm 4 crossover$_C(C_1, C_2)$ shows the 1-point crossover with a quantum random number.

Algorithm 3: Algorithm $crossover\left(P_{tmp}\right)$: Calculate crossover for the current population

INPUT: P_{tmp} *//initial population (with PSIZEchromosomes)*

OUTPUT: P_{new} *//interim population for crossover*

```
1  begin:
2      P_new = ∅    //initialize new population
3      while(P_tmp ≠ ∅)
4          C1, C2 = remove(P_tmp)    //Two chromosomes from current population
5          P_new = P_new ∪ crossover_C(C1, C2)
6      return P_new
7  end
```

Algorithm 4: Algorithm $crossover_C(C_1, C_2)$: Calculate 1-point crossover.

INPUT: C_1, C_2 //*two chromosomes from the current population*

OUTPUT: C_1' , C_2' //*new crossover chromosomes*

```
1 begin:
2     p = QRNG(0...127)//a quantum random number from QRNG within the range
3     C1' = C1[0...(p − 1)] + C2[(p)...127]
4     C2' = C2[0...(p − 1)] + C1[(p)...127]
5     return (C1' , C2')
6 end
```

Mutation: mutate$\left(P_{tmp}\right)$

We are using the mutation operation of a GA [62]. In the GA, mutations can be used to maintain diversity among the chromosomes in a population. A mutation has not been incorporated into the population as a whole, it is measured by a mutation probability that is assigned to each individual according to their fitness value. A random number is generated between 0 and 1 and the mutation probability (k) of our proposed work is 0.95 (see (5)). If the random number is greater than the (k) probability, the values will be generated by QRNG.

The mutation chromosome is calculated as follows:

$$mutate\_c(C) = \begin{cases} QRNG(128), Q_{rn}(0,1) > k \mid \ k = 0.95 \\ C, otherwise \end{cases} \tag{6}$$

where:

$mutate\_c(C)$—Population chromosome after mutation

C—Chromosome of initial population (P_{init})

$Q_{rn}(n_1, n_2)$—a random number from QRNG within the range $n_1 \leq Q_{rn}(n_1, n_2) \leq n_2$

The process will proceed to RMAX generation and the population for the next iteration will be as follows:

$$P_{tmp} = mutate\left(crossover\left(P_{tmp}\right)\right) \tag{7}$$

Algorithm 5 shows the $mutate\_c(C)$ function to calculate the mutation for the current population.

Algorithm 5: Algorithm $mutate\_c\left(P_{tmp}\right)$: Calculate mutation for the current population.

INPUT: P_{tmp} //*initial population (with PSIZEchromosomes)*

OUTPUT: P_{new}//*interim population for mutation*

```
1 begin:
2     P_new = ∅    //initialize new population
3     while(P_tmp ≠ ∅)
4         C = remove(P_tmp)    //chromosome from current population
5         P_new = P_new ∪ mutate_C(C)
6     return P_new
7 end
```

4.2.3. Algorithm $fitness\_GA(C)$

Algorithm 6 shows pseudo-code of the proposed algorithm for $fitness\_GA(C)$ and the calculation of each of the chromosomes fitness value.

Algorithm 6: Algorithm ***fitness_GA*(*C*)**: Calculate fitness value for a given chromosome (128-bit sequence).

INPUT: C *//chromosome (128-bit sequence)*

OUTPUT: $Fitness$ *//fitness value*

```
begin:
    R = C[0...63]        //least significant 64 bits of C
    L = C[64...127]      //most significant 64 bits of C
    H_dist = Hamming_distance(R , L)
    JW_dist = Jaro_Winkler_distance (R , L)
    L_dist = Levenshtein_distance (R , L)
    Fitness = H_dist + JW_dist + L_dist
    return Fitness
end
```

The current chromosome splits into two parts.
Chromosome: (Split the C into two halves R and L)

$$R = C[0 \ldots 63] \tag{8}$$

$$L = C[64 \ldots 127] \tag{9}$$

Different distance values will be calculated using these two parts, leading to the fitness value, $Fitness$. The highest fitness value is the optimum to optimize chromosomes.

4.2.4. Calculation of Distance Values: H_{dist}, $\mathrm{JW_{dist}}$, $\mathrm{L_{dist}}$

$Fitness$ calculates the fitness function for the QuantumGS-box as follows:

$$Fitness = (H_{dist} + JW_{dist} + L_{dist}) \tag{10}$$

where:
H_{dist}—Hamming Distance between R and L: $H_{dist}(R, L)$
JW_{dist}—Jaro-Winkler Distance between R and L: $JW_{dist}(R, L)$
L_{dist} —Levenshtein Distance between R and L: $L_{dist}(R, L)$

Hamming Distance ($\mathrm{H_{dist}}$)

$\mathrm{H_{dist}}(BS_1\ ,\ BS_2)$, the hamming distance [63] between two bit sequences BS_1 and BS_2 is calculated as the number of bit positions $BS_{1,j}$ in BS_1 that are different to those $BS_{2,j}$ in BS_2, divided by the chromosome length:

$$H_{dist}(BS_1, BS_2) = \frac{\left|\ \left\{BS_{1,j} \ \middle|\ BS_{1,j} \neq BS_{2,j} \wedge (0 \leq j < len(BS_1))\right\}\ \right|}{len(BS_1)} \tag{11}$$

Jaro-Winkler Distance ($\mathrm{JW_{dist}}$)

The first step in calculating $\mathrm{JW_{dist}}(BS_1\ ,\ BS_2)$, the Jaro-Winkler Distance between BS_1 and BS_2, is to obtain the Jaro similarity $\mathrm{JS}(BS_1\ ,\ BS_2)$, score between BS_1 and BS_2.

The $JS(BS_1, BS_2)$ score is 0 if the strings do not match at all, and 1 if they are an exact match. JS is calculated as follows:

$$JS(BS_1, BS_2) = \begin{cases} 0, if\ m = 0 \\ \frac{1}{3}\left(\frac{m}{|BS_1|} + \frac{m}{|BS_2|} + \frac{m-t}{m}\right), otherwise \end{cases} \tag{12}$$

where:
$|BS_i|$—is the length of the bitstring BS_i

m—the number of matching bits

t—the number of transpositions

The Jaro–Winkler similarity (JWS) uses a con for a more specific similarity with a defined length len and is calculated as follows:

$$JWS(BS_1, BS_2) = JS(BS_1, BS_2) + len \times con(1 - JS(BS_1, BS_2)) \quad (13)$$

where:

JS is the Jaro similarity between bitstring; BS_1 and BS_2

len—the length of common prefix at the start of the string with up to a maximum of 4 characters

The final JW_{dist} is

$$JW_{dist}(BS_1, BS_2) = 1 - JWS(BS_1, BS_2) \quad (14)$$

Levenshtein Distance (L_{dist})

The Levenshtein distance L_{dist} (BS_1, BS_2) between two bitstring BS_1 and BS_2 is calculated as follows:

$$L_{dist}(BS_1, BS_2) = \begin{cases} |BS_1| & if\ |BS_2| = 0, \\ |BS_2| & if\ |BS_1| = 0, \\ L_{dist}(tail(BS_1),\ tail(BS_2)) & if\ BS_1[0] = BS_2[0], \\ 1 + min \begin{cases} L_{dist}(tail(BS_1), BS_2) \\ L_{dist}(BS_1,\ tail(BS_2) \\ L_{dist}(tail(BS_1),\ tail(BS_2)) \end{cases} & Otherwise \end{cases} \quad (15)$$

The proposed algorithm generates different QuantumGS-boxes based on different generations and keys. Table 2 shows the QuantumGS-box using the 75th generation of the GA and the user defined key—"The QuantumGS-box".

Table 2. QuantumGS-box generated using the 75-th generation of the GA.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	1	67	2B	FE	D7	AB	25
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	F6	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	D9	8D	31	15
3	4	C7	23	C3	18	96	5	9A	7	12	80	E2	EB	27	B2	75
4	9	83	2C	1A	1B	6E	5A	A0	52	3B	D6	72	29	E3	2F	8A
5	53	D1	1D	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	2	7F	50	B3	9F	42
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	B	DB
A	E0	32	3A	A	49	6	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	84	D5	4E	A9	6C	56	F4	EA	65	7A	AE	8
C	BA	78	F8	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	3C
D	70	3E	B5	66	48	3	D8	E	61	35	57	B9	86	C1	B0	9E
E	E1	BF	98	11	69	76	8E	94	9B	1E	87	E9	CE	8C	28	DF
F	A8	A1	89	D	71	E6	55	68	41	99	2D	F	0	54	BB	16

5. Result Analysis

Results are drawn from the strength of S-box parameters such as nonlinearity, bit-independence, avalanche properties including differential and linear approximation analysis. There are several tools to evaluate the performance analysis of cryptographic properties of S-box. The most common tools are MATLAB, Sage [64], SET [65] and the S-BOX performance analysis [66] tool. We evaluated our results using SET and the S-BOX performance tool.

SET-UP experiments and Tools

Hardware: Quantis [29]—A USB-based Quantum random number generators developed by IDQ Its general specifications include—Random bit rate 1:4 Mbit/s ± 10% (Quantis-USB-4M), Thermal noise contribution: <1% (Fraction of random bits arising from thermal noise), Storage temperature: −25 to + 85 °C, USB specification 2.0 and Power Via USB port

Computer—Processor: Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz, RAM: 8.00 GB, System type: 64-bit operating system, x64-based processor

Software: Java—Netbeans with JDK 1.8.0

S-box Testing Tools: SET [65] and S-BOX performance analysis [66]

5.1. Bijectivity Property

An S-box of n × n size is said to be bijective if it has all possible output values from interval [0,2^{n-1}]. If $f_i(1 \leq i \leq n)$ is a Boolean function of an S-box [67], it satisfies

$$\mathrm{wt}\left(\sum_{i=1}^{n} a_i f_i\right) = 2^{n-1} \tag{16}$$

where $a_i \in \{0,1\}, (a_1 a_2 \ldots a_n) \neq (0,0 \ldots 0)$ and wt(x) is the hamming weight, which indicates the number of 1's in a given vector. f_i is to be balanced between 0 and 1. The proposed QuantumGS-box generates the n × n matrix of unique values, as shown in Table 2.

5.2. Non-Linearity

An S-box is strong if it uses a Boolean function with a high non-linearity value. The non-linearity [68] N_f of a Boolean function f(x) is

$$N_f = 2^{n-1}\left(1 - 2^{-n}\max\left|WS_f(\omega)\right|\right) \tag{17}$$

where $WS_f(\omega)$, the Walsh spectrum of function f is defined as

$$WS_f(\omega) = \sum_{\omega \in GF(2^n)}^{n} (-1)^{f(x)+x\cdot\omega} \tag{18}$$

where, $x \cdot \omega$ represents the dot-product of x and ω

Table 3 shows the nonlinearity of different S-boxes generated by different generations. The speed illustrates the time taken to generate the dynamic S-box bits based on the related generations.

Table 3. Nonlinearity of the QuantumGS-box with three different generation and their generation time.

Generation 50, Time: 3.238 s								
Nonlinearity	108	110	112	112	112	110	110	110
Generation 75, Time: 4.560 s								
Nonlinearity	106	108	108	110	108	110	110	110
Generation 100, Time: 4.874 s								
Nonlinearity	112	108	108	108	108	108	108	110

Table 4 presents a comprehensive comparison of our proposed S-box security analysis of nonlinearity. The sample of QuantumGS-box (75th generation), generated by the proposed method is compared with relevant dynamic S-boxes published in the last few years in different category. In the proposed work, we achieve a 110 as the maximum value of nonlinearity with an average of 108.75. Figure 7 shows the graphical representation of nonlinearity comparisons.

Table 4. Comparison of QuantumGS-box with different research studies in terms of nonlinearity.

Design Methodology	Ref.	Non-Linearity		
		Min	Max	Avg
Algebraic	[69]	106	108	107
	[70]	106	108	107.25
	[71]	104	108	106.8
	[72]	102	111	106.5
	[8]	106	110	107.75
	[73]	106	108	106.5
Chaos Based Design	[74]	108	110	109.5
	[75]	104	110	106.3
	[76]	108	112	109.25
	[77]	100	106	104
	[78]	100	110	103.8
	[79]	100	108	105
Others	[13]	99	108	103.5
	[80]	102	110	106.5
	[59]	102	110	107
QuantumGS-box 75 Gen	this work	106	110	108.75

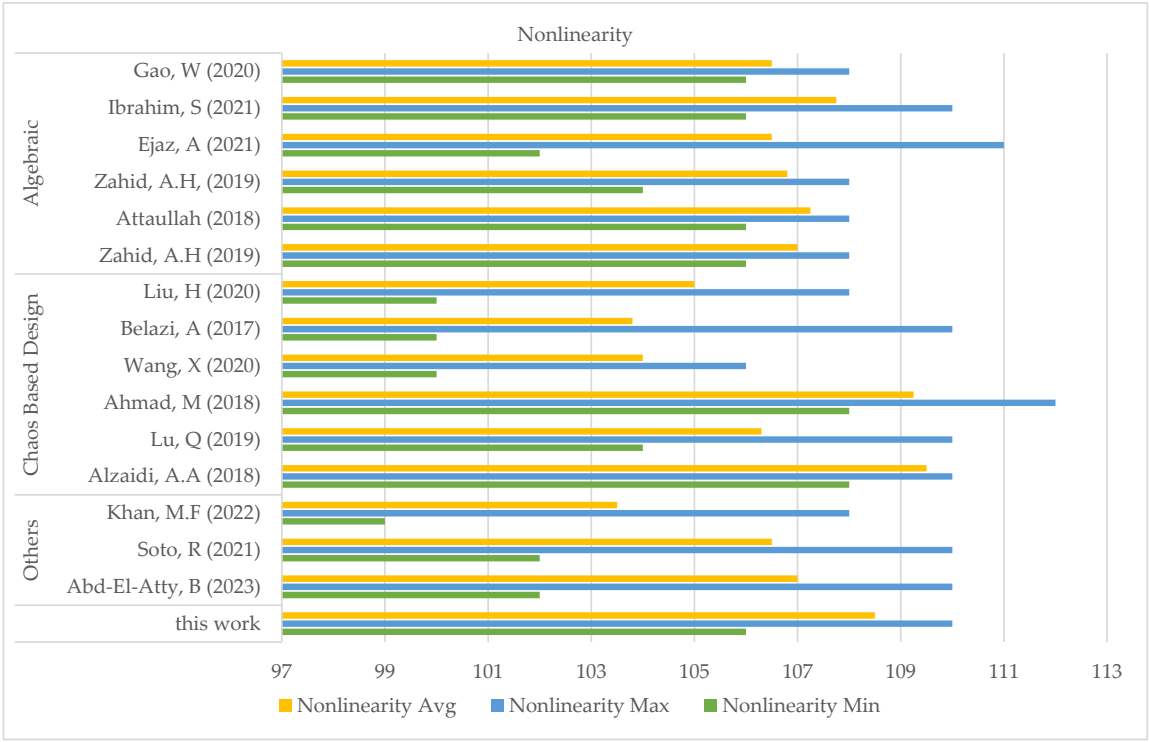


Figure 7. Comparison of QuantumGS-box(this work) with different research studies in terms of nonlinearity. Zahid, A.H (2019)—[69]; Attaullah (2018)—[70]; Zahid, A.H, (2019)—[71]; Ejaz, A (2021)—[72]; Ibrahim, S (2021)—[8]; Gao, W (2020)—[73]; Al-zaidi, A.A (2018)—[74]; Lu, Q (2019)—[75]; Ahmad, M (2018)—[76]; Wang, X (2020)—[77]; Belazi, A (2017)—[78]; Liu, H (2020)—[79]; Khan, M.F (2022)—[13]; Soto, R (2021)—[80]; Abd-El-Atty, B (2023)—[59].

5.3. Bit-Independence Criteria—Strict Avalanche Criteria (BIC-SAC)

Bit-Independence Criteria (BIC) tests measure the correlation between each pair of output bits when one input bit changes in response to the change in the output bit. In this test, the diagonal values are excluded from the output matrix of n × n dimensions.

If a Boolean function satisfies the *Strict Avalanche Criteria* (SAC) [46], half of the output bits should change when there is a change in a bit. Any change in the input vector will significantly change the output vector with a probability of ½. Table 5 shows the BIC-SAC values of the proposed QuantumGS-box. The maximum, minimum and average SAC values for the QuantumGS-box are equal to 0.525391, 0.46875, and 0.497070 correspondingly. Based on the results, the proposed QuantumGS-box fulfils the SAC property as the average value for the QuantumGS-box is equal to the desired value of SAC (0.5).

Table 5. BIC-SAC of QuantumGS-box.

0	**0.511719**	**0.503906**	**0.525391**	**0.513672**	**0.486328**	**0.46875**	**0.509766**
0.511719	0	0.511719	0.496094	0.498047	0.507813	0.507813	0.515625
0.503906	0.511719	0	0.527344	0.501953	0.507813	0.486328	0.505859
0.525391	0.496094	0.527344	0	0.519531	0.517578	0.503906	0.503906
0.513672	0.498047	0.501953	0.519531	0	0.519531	0.515625	0.5
0.486328	0.507813	0.507813	0.517578	0.519531	0	0.509766	0.488281
0.46875	0.507813	0.486328	0.503906	0.515625	0.509766	0	0.517578
0.509766	0.515625	0.505859	0.503906	0.5	0.488281	0.517578	0

5.4. Linear Approximation Probability (LAP)

The LAP assesses the security of S-boxes against linear cryptanalysis. An S-box provides diffusion and confusion of bits through linear mappings between inputs and outputs. The LAP of an event determines a maximum imbalance [48,65].

$$LAP = \max_{p_x, q_x \neq 0} \left| \frac{|\{ x \mid (x \in R) \wedge (x \cdot p_x = S(x) \cdot q_x) \}|}{2^n} - \frac{1}{2} \right| \tag{19}$$

where:

p_x and q_x are the input and output values respectively and $R = \{1, 2, 3, 4, \ldots, 255\}$

A small linear probability in an S-box makes the box very resistant to linear cryptanalysis. In our proposed work, we achieve a LAP value of 0.1015. Table 6 compares the maximum LP of different researchers with the proposed work. Figure 8 shows the graphical representation of the comparison data.

Table 6. Comparison of QuantumGS-box with different research studies in terms of LAP.

Design Methodology	**Ref.**	**Linear Approximation Probability (LAP)**
Algebraic	[69]	0.1560
	[70]	0.1094
	[71]	0.1400
	[72]	0.1090
	[8]	0.1172
	[73]	0.1250

Table 6. *Cont.*

Design Methodology	Ref.	Linear Approximation Probability (LAP)
Chaos Based Design	[74]	0.1328
	[75]	0.1250
	[76]	0.1250
	[77]	0.1328
	[78]	0.1250
	[79]	0.1250
Others	[13]	0.1406
	[80]	0.1484
	[59]	0.1172
QuantumGS-box 75 Gen	this work	0.1015

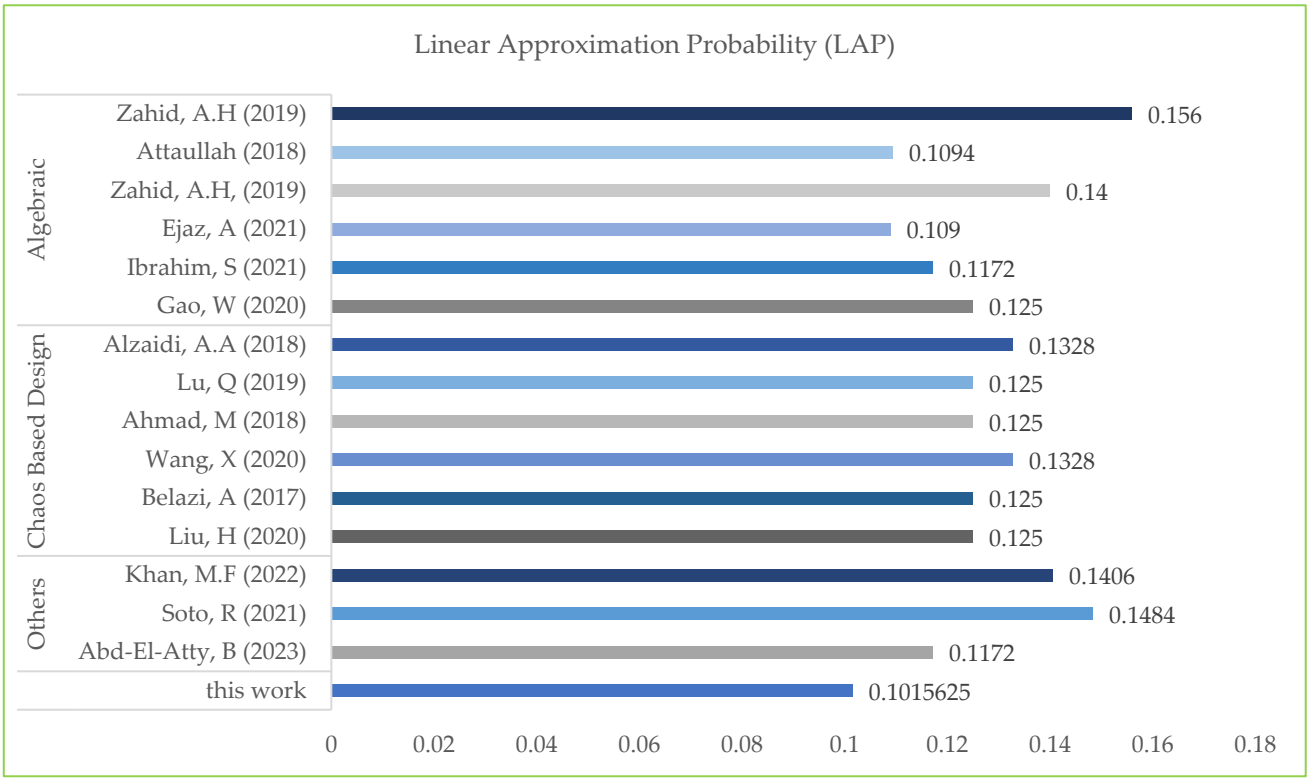


Figure 8. Comparison of the QuantumGS-box with different research studies in terms of LAP. Zahid, A.H (2019)—[69]; Attaullah (2018)—[70]; Zahid, A.H, (2019)—[71]; Ejaz, A (2021)—[72]; Ibrahim, S (2021)—[8]; Gao, W (2020)—[73]; Alzaidi, A.A (2018)—[74]; Lu, Q (2019)—[75]; Ahmad, M (2018)—[76]; Wang, X (2020)—[77]; Belazi, A (2017)—[78]; Liu, H (2020)—[79]; Khan, M.F (2022)—[13]; Soto, R (2021)—[80]; Abd-El-Atty, B (2023)—[59].

5.5. Differential Approximation Probability (DP)

An effective way to assess S-box resistance to differential attacks is to use the differential approximation probability (DP) [48,81]. DP indicates the probability that a particular change in output bits will occur due to a change in input bits.

The *DP* is calculated as follows

$$DP(\Delta x \rightarrow \Delta y) = \left(\frac{|\{\, x \mid \ (x \in X) \ \wedge \ ((S(x) \oplus S(x \oplus \Delta x)) = \Delta y)\}|}{2^n} \right) \tag{20}$$

where

x represents the set of all possible input values, 2n is a total number of all the elements in the S-box with a small differential value is strongly resistant to differential cryptoanalysis. In our proposed work, we achieve a DAP value of 0.03125.

5.6. Balanced Output

An S-box with n input bits and m output bits, $m \leq n$, is balanced if each output occurs $2n-m$ times. For the S-box to be balanced [46], it should have the same number of 0's and 1's. The result from the SET [65] tool assessed that the QuantumGS-box is balanced.

6. Conclusions

Our research concludes that the quantum random number-based genetic algorithm can generate dynamic QuantamGS-boxes with different random S-box values at different arbitrary positions for cloud-based storage. The dynamic feature of the algorithm based on bit-shuffling with the operations of a genetic algorithm also makes it more resilient to modern cryptographic attacks. In our proposed work, we achieve the lowest LAP value when compared to other researches. QuantumGS-boxes generated by the proposed algorithm assures nonlinear. The QuantumGS-box also has low differential and linear approximation properties in addition to satisfying bijectivity cryptography properties, making it resistant to differential and linear attacks. Low differential and linear approximation properties makes guessing the algorithm's output from its inputs difficult and resistant to attacks that seek to exploit patterns in the input or output of the cryptographic system. Our proposed QuantumGS-box ensures that cloud-based storage over will be more resilient to various cryptographic attacks.

The future work of this research focuses on the proposed QuantumGSbox for the LoRaWAN IoT security symmetric algorithm and can be used to enhance the IoT security. LoRaWAN is a low-power wide-area network that connects devices with long-range communication over a low bit rate. This protocol is widely used in IoT communication. The encryption process of the LoRaWAN using the QuantumGS-box encrypts the user's data. The user sends the encrypted data over the high-speed network on IoT cloud storage and receives the same encrypted data.

Author Contributions: Methodology, A.S., A.T. and R.K.; Software, A.S.; Validation, R.K.; Data curation, A.S.; Writing—original draft, A.S., A.T. and R.K.; Writing—review & editing, A.S., A.T. and R.K.; Visualization, A.S. and R.K.; Supervision, A.T. and R.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research work received funding by University of Hertfordshire, United Kingdom.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhang, X.; Li, W.; Hu, H.; Dutta, N.K. High-speed all-optical encryption and decryption based on two-photon absorption in semiconductor optical amplifiers. *J. Opt. Commun. Netw.* **2015**, *7*, 276–285. [CrossRef]
2. Stallings, W. *Cryptography and Network Security: Principles and Practices*, 6th ed.; Prentice Hall Press: Upper Saddle River, NJ, USA, 2013.
3. May, L.; Henricksen, M.; Millan, W.; Carter, G.; Dawson, E. Strengthening the key schedule of the AES. *Lect. Notes Comput. Sci.* **2002**, *2384*, 226–240. [CrossRef]
4. Hughes, R.; Nordholt, J. *Strengthening the Security Foundation of Cryptography with Whitewood's Quantum-Powered Entropy Engine*; Whitewood Encryption Systems, Inc.: Boston, MA, USA, 2016; Available online: http://www.whitewoodencryption.com (accessed on 3 August 2021).
5. Smart, N.P.; Rijmen, V.; Warinschi, B.; Watson, G. *Algorithms, Key Sizes and Parameters Report*; ENISA: Athens, Greece, 2014. Available online: https://www.enisa.europa.eu/publications/algorithms-key-size-and-parameters-report-2014 (accessed on 9 September 2021).
6. Siddiqui, N.; Khalid, H.; Murtaza, F.; Ehatisham-Ul-Haq, M.; Azam, M.A. A novel algebraic technique for design of computational substitution-boxes using action of matrices on galois field. *IEEE Access* **2020**, *8*, 197630–197643. [CrossRef]

7. Alharbi, A.R.; Jamal, S.S.; Khan, M.F.; Gondal, M.A.; Abbasi, A.A. Construction and Optimization of Dynamic S-Boxes Based on Gaussian Distribution. *IEEE Access* **2023**, *11*, 35818–35829. [CrossRef]
8. Ibrahim, S.; Abbas, A.M. Efficient key-dependent dynamic S-boxes based on permutated elliptic curves. *Inf. Sci.* **2021**, *558*, 246–264. [CrossRef]
9. Zahid, A.H.; Tawalbeh, L.; Ahmad, M.; Alkhayyat, A.; Hassan, M.T.; Manzoor, A.; Farhan, A.K. Efficient Dynamic S-Box Generation Using Linear Trigonometric Transformation for Security Applications. *IEEE Access* **2021**, *9*, 98460–98475. [CrossRef]
10. Lu, Q.; Zhu, C.; Deng, X. An Efficient Image Encryption Scheme Based on the LSS Chaotic Map and Single S-Box. *IEEE Access* **2020**, *8*, 25664–25678. [CrossRef]
11. Ibrahim, S.; Abbas, A.M.; Alharbi, A.A.; Albahar, M.A. A New 12-Bit Chaotic Image Encryption Scheme Using a 12 × 12 Dynamic S-Box. *IEEE Access* **2024**, *12*, 37631–37642. [CrossRef]
12. Özkaynak, F. On the effect of chaotic system in performance characteristics of chaos based s-box designs. *Phys. A Stat. Mech. Its Appl.* **2020**, *550*, 124072. [CrossRef]
13. Khan, M.F.; Saleem, K.; Alotaibi, M.; Hazzazi, M.M.; Rehman, E.; Abbasi, A.A.; Gondal, M.A. Construction and Optimization of TRNG Based Substitution Boxes for Block Encryption Algorithms. *Comput. Mater. Contin.* **2022**, *73*, 2679–2696. [CrossRef]
14. Ibrahim, H.; Ozkaynak, F. A Substitution-Box Structure Based on Crowd Supply Infinite Noise TRNG. In Proceedings of the 9th International Symposium on Digital Forensics and Security, ISDFS, Elazig, Turkey, 28–29 June 2021; Volume 2021. [CrossRef]
15. Kim, G.; Kim, H.; Heo, Y.; Jeon, Y.; Kim, J. Generating Cryptographic S-Boxes Using the Reinforcement Learning. *IEEE Access* **2021**, *9*, 83092–83104. [CrossRef]
16. Yang, M.; Wang, Z.; Meng, Q.; Han, L. Evolutionary design of S-box with cryptographic properties. In Proceedings of the 9th IEEE International Symposium on Parallel and Distributed Processing with Applications Workshops, ISPAW 2011, ICASE 2011, SGH 2011, GSDP 2011, Busan, Republic of Korea, 26–28 May 2011; p. 12. [CrossRef]
17. Kadhim, A.F.; Kamal, Z.A. Generating dynamic S-BOX based on Particle Swarm Optimization and Chaos Theory for AES. *Iraqi J. Sci.* **2018**, *59*, 1733–1745. [CrossRef]
18. Sahmoud, S.; Elmasry, W.; Shadi, A. Enhancement the Security of AES Against Modern Attacks by Using Variable Key Block Cipher. *Int. Arab J. E-Technol.* **2013**, *3*, 17–26.
19. Oracle. SecureRandom. 2020. Available online: https://docs.oracle.com/javase/8/docs/technotes/guides/security/StandardNames.html#SecureRandom (accessed on 23 June 2020).
20. Saha, R.; Geetha, G.; Kumar, G.; Kim, T.H. RK-AES: An Improved Version of AES Using a New Key Generation Process with Random Keys. *Secur. Commun. Netw.* **2018**, *2018*, 9802475. [CrossRef]
21. Jane, M.; Bruno, H.; Moulds, R.; Nino, W.; Anthony, F. *Quantum-Safe Security Working Group Quantum Random Number Generators*; Cloud Security Alliance: Bellingham, WA, USA, 2016.
22. Quantique, I.D. What Is the Q in QRNG? 2020. Available online: https://www.idquantique.com/random-number-generation/overview/ (accessed on 7 July 2020).
23. Quantique, I.D. Understanding Quantum Cryptography. 2020. Available online: https://www.idquantique.com/quantum-safe-security/quantum-key-distribution/ (accessed on 7 July 2020).
24. Herrero-Collantes, M.; Garcia-Escartin, J.C. Quantum random number generators. *Rev. Mod. Phys.* **2017**, *89*, 015004. [CrossRef]
25. Cao, Z.; Zhou, H.; Yuan, X.; Ma, X. Source-independent quantum random number generation. *Phys. Rev. X* **2016**, *6*, 011020. [CrossRef]
26. Liu, J.; Yang, J.; Li, Z.; Su, Q.; Huang, W.; Xu, B.; Guo, H. 117 Gbits/s Quantum Random Number Generation with Simple Structure. *IEEE Photonics Technol. Lett.* **2017**, *29*, 283–286. [CrossRef]
27. Siswanto, M.; Rudiyanto, B. Designing of quantum random number generator (QRNG) for security application. In Proceedings of the 2017 3rd International Conference on Science in Information Technology (ICSITech), Bandung, Indonesia, 25–26 October 2017; pp. 273–277. [CrossRef]
28. Quantique, I.D. Gaming-and-Lotteries. Available online: https://www.idquantique.com/random-number-generation/applications/gaming-and-lotteries/ (accessed on 7 July 2020).
29. IDQ. Quantis-Random-Number-Generator. Available online: https://www.idquantique.com/random-number-generation/products/quantis-random-number-generator (accessed on 7 July 2020).
30. Picek, S.; Marchiori, E.; Batina, L.; Jakobovic, D. Combining evolutionary computation and algebraic constructions to find cryptography-relevant boolean functions. *Lect. Notes Comput. Sci.* **2014**, *8672*, 822–831. [CrossRef]
31. Katoch, S.; Chauhan, S.S.; Kumar, V. A review on genetic algorithm: Past, present, and future. *Multimed. Tools Appl.* **2021**, *80*, 8091–8126. [CrossRef]
32. Wang, Y.; Wong, K.W.; Li, C.; Li, Y. A novel method to design S-box based on chaotic map and genetic algorithm. *Phys. Lett. Sect. A Gen. At. Solid State Phys.* **2012**, *376*, 827–833. [CrossRef]
33. Wang, Y.; Zhang, Z.; Zhang, L.Y.; Feng, J.; Gao, J.; Lei, P. A genetic algorithm for constructing bijective substitution boxes with high nonlinearity. *Inf. Sci.* **2020**, *523*, 152–166. [CrossRef]
34. Nitaj, A.; Susilo, W.; Tonien, J. A New Improved AES S-box with Enhanced Properties. *Lect. Notes Comput. Sci.* **2020**, *12248*, 125–141. [CrossRef]
35. Daemen, J.; Rijmen, V. *The Design of Rijndael*; Springer: Berlin/Heidelberg, Germany, 2002; ISBN 3540425802.

36. Gullasch, D.; Bangerter, E.; Krenn, S. Cache games Bringing access-based cache attacks on AES to practice. In Proceedings of the IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 2–25 May 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 490–505.
37. Biryukov, A.; Khovratovich, D. Related-key cryptanalysis of the full AES-192 and AES-256. In *Advances in Cryptology ASIACRYPT Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 1–18. ISBN 3642103650.
38. Juremi, J.; Sulaiman, S.; Saad, N.H.M. A survey on various dynamic S-box implementation in block cipher encryption algorithm. *J. Appl.* **2019**, *3*, 2–5.
39. Gupta, M.D.; Chauhan, R.K. Secure image encryption scheme using 4D-Hyperchaotic systems based reconfigurable pseudo-random number generator and S-Box. *Integration* **2021**, *81*, 137–159. [CrossRef]
40. Zhao, M.; Liu, H.; Niu, Y. Batch generating keyed strong S-Boxes with high nonlinearity using 2D hyper chaotic map. *Integration* **2023**, *92*, 91–98. [CrossRef]
41. Arrag, S.; Hamdoun, A.; Tragha, A.; Khamlich Salah, E. Implementation of stronger AES by using dynamic S-box dependent of master key. *J. Theor. Appl. Inf. Technol.* **2013**, *53*, 196–204.
42. Alamsyah Improving the Quality of AES S-box by Modifications Irreducible Polynomial and Affine Matrix. In Proceedings of the 2020 5th International Conference on Informatics and Computing, ICIC 2020, Gorontalo, Indonesia, 3–4 November 2020. [CrossRef]
43. Alamsyah; Prasetiyo, B.; Ardian, M.N. Enhancement security AES algorithm using a modification of transformation ShiftRows and dynamic S-box. *J. Phys. Conf. Ser.* **2020**, *1567*, 032025. [CrossRef]
44. Manjula, G.; Mohan, H.S. Constructing key dependent dynamic S-Box for AES block cipher system. In Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2016, Bengaluru, India, 21–23 July 2016; pp. 613–617. [CrossRef]
45. Agarwal, P.; Singh, A.; Kilicman, A. Development of key-dependent dynamic S-Boxes with dynamic irreducible polynomial and affine constant. *Adv. Mech. Eng.* **2018**, *10*, 1–18. [CrossRef]
46. Jacob, G.; Murugan, A.; Viola, I. Towards the Generation of a Dynamic Key-Dependent S-Box to Enhance Security. *IACR Cryptol. ePrint Arch.* **2015**, *2015*, 92.
47. Cassal-Quiroga, B.B.; Campos-Cantón, E. Generation of Dynamical S-Boxes for Block Ciphers via Extended Logistic Map. *Math. Probl. Eng.* **2020**, *2020*, 2702653. [CrossRef]
48. Mahmood Malik, M.S.; Ali, M.A.; Khan, M.A.; Ehatisham-Ul-Haq, M.; Shah, S.N.M.; Rehman, M.; Ahmad, W. Generation of Highly Nonlinear and Dynamic AES Substitution-Boxes (S-Boxes) Using Chaos-Based Rotational Matrices. *IEEE Access* **2020**, *8*, 35682–35695. [CrossRef]
49. Aguirre, H.; Okazaki, H.; Fuwa, Y. An evolutionary multiobjective approach to design highly non-linear Boolean functions. In Proceedings of the GECCO 2007: Genetic and Evolutionary Computation Conference, New York, NY, USA, 9–13 July 2007; pp. 749–756. [CrossRef]
50. Picek, S.; Jakobovic, D.; Golub, M. Evolving cryptographically sound boolean functions. In Proceedings of the GECCO 2013 Genetic and Evolutionary Computation Conference Companion, Amsterdam, The Netherlands, 6–10 July 2013; pp. 191–192. [CrossRef]
51. Kalaiselvi, K.; Kumar, A. A Novel Method to Design S-box Based on Genetic Algorithm and Particle Swarm Optimizationin AES-128 Cryptosystem. 2018, 118, 1443–1457. *Int. J. Pure Appl. Math.* **2018**, *118*, 1443–1457.
52. Alsaif, H.; Guesmi, R.; Kalghoum, A.; Alshammari, B.M.; Guesmi, T. A Novel Strong S-Box Design Using Quantum Crossover and Chaotic Boolean Functions for Symmetric Cryptosystems. *Symmetry* **2023**, *15*, 833. [CrossRef]
53. Iavich, M.; Kuchukhidze, T.; Okhrimenko, T.; Dorozhynskyi, S. Novel Quantum Random Number Generator for Cryptographical Applications. In Proceedings of the 2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T), Kharkiv, Ukraine, 6–9 October 2020; pp. 727–732. [CrossRef]
54. Shaw, G.; Sivaram, S.R.; Prabhakar, A. Quantum Random Number Generator with One and Two Entropy Sources. In Proceedings of the National Conference on Communications (NCC), Bangalore, India, 20–23 February 2019; IEEE: Piscataway, NJ, USA, 2019; p. 1. [CrossRef]
55. Mogos, G. Quantum Random Number Generator vs. In Random Number Generator. In Proceedings of the IEEE International Conference on Communications, Kuala Lumpur, Malaysia, 22–27 May 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 423–426.
56. IDQ. *Quantum versus Classical Random Number Generators*; ID QUANTIQUE SA: Geneva, Switzerland, 2020.
57. Albrecht, M.R. Algebraic Techniques in Cryptanalysis. *Ecrypt2[Ppt]* **2011**, *3*, 120–141.
58. Bardeh, N.G.; Rønjom, S. Practical attacks on reduced-round AES. *Lect. Notes Comput. Sci.* **2019**, *11627*, 297–310. [CrossRef]
59. Abd-El-Atty, B. Efficient S-box construction based on quantum-inspired quantum walks with PSO algorithm and its application to image cryptosystem. *Complex Intell. Syst.* **2023**, *9*, 4817–4835. [CrossRef]
60. Kalaiselvi, K.; Kumar, A. Enhanced AES cryptosystem by using genetic algorithm and neural network in S-box. In Proceedings of the 2016 IEEE International Conference on Current Trends in Advanced Computing ICCTAC 2016, Bangalore, India, 10–11 March 2016. [CrossRef]
61. Zhu, D.; Zhang, M.; Tong, X.; Wang, Z. A Novel S-box Optimization Method Based on Immune Genetic Algorithm. In Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS, Beijing, China, 21–23 October 2022; p. 32. [CrossRef]

62. Zhu, D.; Tong, X.; Zhang, M.; Wang, Z. A new s-box generation method and advanced design based on combined chaotic system. *Symmetry* **2020**, *12*, 2087. [CrossRef]
63. Community, T.S. Hamming. Available online: https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.hamming.html (accessed on 9 July 2020).
64. Team, T.S.D. Sage. Available online: https://doc.sagemath.org/html/en/reference/cryptography/index.html (accessed on 14 November 2022).
65. Picek, S.; Batina, L.; Jakobović, D.; Ege, B.; Golub, M. S-box, SET, Match: A Toolbox for S-box Analysis. In Proceedings of the 8th IFIP International Workshop on Information Security Theory and Practice (WISTP), Heraklion, Crete, Greece, 30 June–2 July 2014; pp. 140–149. [CrossRef]
66. YongWang, D. S-Box Performance Analysis. Available online: http://42.192.236.76/Login (accessed on 17 March 2023).
67. Ahmad, M.; Haleem, H.; Khan, P.M. A new chaotic substitution box design for block ciphers. In Proceedings of the 2014 International Conference on Signal Processing and Integrated Networks, SPIN 2014, Noida, India, 20–21 February 2014; pp. 255–258. [CrossRef]
68. Ahmad, M.; Mittal, N.; Garg, P.; Maftab Khan, M. Efficient cryptographic substitution box design using travelling salesman problem and chaos. *Perspect. Sci.* **2016**, *8*, 465–468. [CrossRef]
69. Zahid, A.H.; Arshad, M.J.; Ahmad, M. A novel construction of efficient substitution-boxes using cubic fractional transformation. *Entropy* **2019**, *21*, 245. [CrossRef]
70. Attaullah; Jamal, S.S.; Shah, T. A Novel Algebraic Technique for the Construction of Strong Substitution Box. *Wirel. Pers. Commun.* **2018**, *99*, 213–226. [CrossRef]
71. Zahid, A.H.; Arshad, M.J. An innovative design of substitution-boxes using cubic polynomial mapping. *Symmetry* **2019**, *11*, 437. [CrossRef]
72. Ejaz, A.; Shoukat, I.A.; Iqbal, U.; Rauf, A.; Kanwal, A. A secure key dependent dynamic substitution method for symmetric cryptosystems. *PeerJ Comput. Sci.* **2021**, *7*, 587. [CrossRef]
73. Gao, W.; Idrees, B.; Zafar, S.; Rashid, T. Construction of Nonlinear Component of Block Cipher by Action of Modular Group PSL(2, Z) on Projective Line PL(GF(2 8)). *IEEE Access* **2020**, *8*, 136736–136749. [CrossRef]
74. Alzaidi, A.A.; Ahmad, M.; Ahmed, H.S.; Solami, E. Al Sine-Cosine Optimization-Based Bijective Substitution-Boxes Construction Using Enhanced Dynamics of Chaotic Map. *Complexity* **2018**, *2018*, 9389065. [CrossRef]
75. Lu, Q.; Zhu, C.; Wang, G. A Novel S-Box Design Algorithm Based on a New Compound Chaotic System. *Entropy* **2019**, *21*, 1004. [CrossRef]
76. Ahmad, M.; Doja, M.N.; Beg, M.M.S. ABC Optimization Based Construction of Strong Substitution-Boxes. *Wirel. Pers. Commun.* **2018**, *101*, 1715–1729. [CrossRef]
77. Wang, X.; Yang, J. A novel image encryption scheme of dynamic S-boxes and random blocks based on spatiotemporal chaotic system. *Optik* **2020**, *217*, 164884. [CrossRef]
78. Belazi, A.; El-Latif, A.A.A. A simple yet efficient S-box method based on chaotic sine map. *Optik* **2017**, *130*, 1438–1444. [CrossRef]
79. Liu, H.; Kadir, A.; Xu, C. Cryptanalysis and constructing S-Box based on chaotic map and backtracking. *Appl. Math. Comput.* **2020**, *376*, 125153. [CrossRef]
80. Soto, R.; Crawford, B.; Molina, F.G.; Olivares, R. Human Behaviour Based Optimization Supported with Self-Organizing Maps for Solving the S-Box Design Problem. *IEEE Access* **2021**, *9*, 84605–84618. [CrossRef]
81. Khan, M.F.; Saleem, K.; Shah, T.; Hazzazi, M.M.; Bahkali, I.; Shukla, P.K. Block Cipher's Substitution Box Generation Based on Natural Randomness in Underwater Acoustics and Knight's Tour Chain. *Comput. Intell. Neurosci.* **2022**, *2022*, 8338508. [CrossRef] [PubMed]